

Chapter 7

Implicit Collaborations Across the DOE Mission Sciences

The advanced computational biology simulations described in this white paper will require computer performance well beyond what is currently available, but computational speed alone will not ensure that the computer is useful for any specific simulation method. Several other factors are critical including the size of primary system memory (RAM), the size and speed of secondary storage, and the overall architecture of the computer. This will undoubtedly be true for global climate, combustion, and other basis energy science areas.

Many parallel processing computer architectures have been developed over the

years, but the dominant parallel architecture that has emerged is the distributed memory, multiple instruction-multiple data architecture (MIMD), which consists of a set of independent processors with their own local RAM memory interconnected by some sort of communication channel. Such an architecture is characterized by the topology and speed of the interconnection network, and by the speed and memory size of the individual processors. All of the current generation of teraFLOP-class computers, including ASCI Red and Blue are of this design.

Computational modeling plays a key role in all scientific understanding. Therefore it is probably not surprising that disparate scientific disciplines share common modeling strategies, algorithmic bottlenecks, and information technology issues, and use computer scientists and applied mathematicians as a common resource. In this Section we emphasize what are the cross-cutting computational problems whose solutions could benefit a large number of scientific applications, or where technologies should be transferred between the computational biology domain and other disciplines.

Just as with the computer hardware, there have been a large number of software programming paradigms developed for parallel computers. A great deal of research has gone into developing software tools to assist in parallel programming or even to automatically parallelize existing single-processor software. Selected parallelization and debugging tools can assist and new programming models such as Object-Oriented programming (using C++ or FORTRAN90) can help hide the details of the underlying computer architecture. At the current time, however,

efficiently using massively parallel computers primarily involves redesigning and rewriting software by hand. This is complicated by the facts that the best serial (single processor) algorithms are often not the best suited for parallel computers and the optimal choice of algorithm often depends on the details of the computer hardware.

The different simulation methods presently used have different requirements of parallel computer hardware. Simulation methods which involve calculating averaged properties from a large number of

smaller calculations which can be individually run on gigaFLOP class processors are most ideally suited for parallelism. These methods include classical and quantum Monte Carlo, as well as global optimization methods. In these simulations a minimal amount of initial data can be sent to each processor which then independently calculates a result that is communicated back to a single processor. By choosing an appropriate size of problem for each single processor (problem granularity), these algorithms will work efficiently on virtually any MIMD computer, including separate computer workstations linked by local-area networks.

The quantum chemical and molecular dynamics methods, in which all processors are applied to the calculation of a single chemical wave function or trajectory, involve much greater challenges to parallelization and involve greater constraints on the parallel computer architecture. Since all processors are involved in a single computation, interprocessor communication must occur. It is the rate of this communication, characterized in term of raw speed (bandwidth) and initialization time (latency) that usually limits the efficient use of parallel computers. The minimal necessary communication rate depends exquisitely on the simulation type, choice of algorithm and problem size. Generally, it is essential software design criteria that as the problem scales to larger size, the ratio of computational operations per communication decrease (or at least remain constant), so that for some problem size, the communication rate will not constitute a bottleneck. Moreover, it is important that the work per processor, or “load balance”, scale evenly so that no processors end up with much larger computational loads and become bottlenecks. In a broad sense, the nature of computational biology simulations—in particular the physical principle that interactions attenuate with distance—will

ensure that scalable parallel algorithms can be developed, albeit at some effort.

Even given the very broad range of simulation methods required by computational biology, it is possible to provide some guidelines for the most efficient computer architectures. Regarding the size of primary memory, it is usually most efficient if a copy of the $(6 \times N)$ set of coordinates describing a timestep of a molecular dynamics simulation or the $(N \times N)$ matrices describing the quantum chemical wavefunction, can be stored on each processing element. For the biological systems of the sort described in this white paper, this corresponds to a minimum of several hundred megabytes of RAM per processor. Moreover, since many of the simulation methods involve the repeated calculation of quantities that could be stored and reused (e.g. two electron integrals in quantum chemistry or interaction lists in molecular dynamics), memory can often be traded for computer operations so that larger memory size will permit even larger simulations. Similarly, general estimates can be made for the minimal interprocessor communication rates. Since the goal of parallel processing is to distribute the effort of a calculation, for tightly coupled methods such as quantum chemical simulations, it is essential that the time to communicating a partial result be less than the time to simply recalculate it. For example, the quantum chemical two-electron integrals require 10-100 floating point operations to calculate, so that they can be usefully sent to other processors only if that requires less than ~100 cycles to communicate to send the 8 or 16 byte result. Assuming gigaFLOP speeds for individual processing elements in the parallel computers, this translates roughly to gigabyte/sec interprocessor communication speeds. (Note that many partial results involve vastly more operations, so that they place a much weaker constraint on the communication rate.

Information Technologies and Database Management. "Biology is an Information Science" and the field is poised to put into practice new information science and data management technologies directly. Two major conferences are emerging within the field of computational biology (ISMB - Intelligent Systems for Molecular Biology; and RECOMB - Research in Computational Biology). Each year, associated workshops focus on how to push new techniques from computer science into use in computational biology. For example, at ISMB-94 a workshop focused on problems involved in integrating biological databases; follow-up workshops in 1995 and 1996 explored CORBA and java as methods toward integration solutions. In 1997, a post-conference workshop focused on issues in accurate, usable annotations of genomes. This year, a pre-conference workshop will explore how to use text-processing and machine-translation methods for building ontologies to support cross-linking between databases about organisms. All of these criteria require ultra-high-speed networks to interconnect students, experimental biologists, and computational biologists and publicly funded data repositories. This community will, for example, benefit directly from every new distributed networking data exchange tool that develops as a result of Internet-II and the high-speed Energy Sciences network.

Data warehousing addresses a fundamental data management issue: the need to transparently query and analyze data from multiple heterogeneous sources distributed across an enterprise or a scientific community. Warehousing techniques have been successfully applied to a multitude of business applications in the commercial world. Although the need for this capability is as vital in the sciences as in business, functional warehouses tailored for specific scientific needs are few and far between. A key technical reason for this discrepancy is that our understanding of the concepts being explored

in an evolving scientific domain change constantly, leading to rapid changes in data representation. When the format of source data changes, the warehouse must be updated to read that source or it will not function properly. The bulk of these modifications involve extremely tedious, low-level translation and integration tasks which typically require the full attention of both database and domain experts. Given the lack of the ability to automate this work, warehouse maintenance costs are prohibitive, and warehouse "up-times" severely restricted. This is the major roadblock to a successful warehouse for scientific data domains. Regardless of whether the scientific domain is genome, combustion, high energy physics, or climate modeling, the underlying challenges for data management are similar, and present in varying degrees for any warehouse. We need to move towards the automation of scientific these tasks. Research will play a vital role in achieving that goal and in scaling warehousing approaches to dynamic scientific domains. Warehouse implementations have an equally important role; they allow one to exercise design decisions, and provide a test-bed that stimulates the research to follow more functional and robust paths.

Ensuring Scalability on Parallel Architectures. In all of the research areas, demonstrably successful parallel implementations must be able to exploit each new generation of computer architectures that will rely on increased number of processors to realize multiple teraflop computing. We see the use of the paradigm/software support tools as an important component of developing effective parallelization strategies that can fully exploit the increased number of processors that will comprise a 100 teraflop computing resource. However, these paradigm/software support libraries have largely been developed on model problems at a finer level of granularity than "real life" computational problems. But it is the "real

life" problems that involve complexity in lengthscales, timescales, and severe scalings of algorithmic kernels that are in need of the next and future generations of multiple teraflop computing. The problems described in this initiative provide for a more realistic level of granularity to investigate the improved use of software support tools for parallel implementations, and may improve the use of high computing resources by the other scientific disciplines, such as climate, combustion, and materials, that need scalable algorithms.

Even when kernels can be identified and parallel algorithms can be designed to solve them, often the implementation does not scale well with the number of processors. Since 100 teraflop computing will only be possible on parallel architectures, problems in scalability is a severe limitation in realizing the goals of the Accelerated Computational Biotechnology Initiative. Lack of scalability often arises from straightforward parallel implementations where the algorithm is controlled by a central scheduler and for which communication among processors is wired to be synchronous. In the computer science community, various paradigms and software library support modules exist for exploring better parallel implementations. These can decompose the requirements of a problem domain into high-level modules, each of which is efficiently and portably supported in a software library that can address issues involving communication, embedding, mapping, etc for a scientific application of interest.

These tools allow different decompositions of a parallel implementation to be rapidly explored, by handling all of the low-level communication for a given platform. However, these paradigms and their supporting software libraries usually are developed in the context of model mathematical applications, which are at a finer level of granularity than "real life"

computational science problems. A unique opportunity exists to use some of the computational science problems described in the previous chapters to refine or redefine the current parallel paradigms currently in use. The outcome of this direction could be broader than the particular scientific application, and may provide insight in how to improve the use of parallel computing resources in general.

Meta Problem Solving Environments. In order to take advantage of teraflop computing power, biological software design must move towards an efficient geographically distributed software paradigm. One novel paradigm is the "Meta Problem Solving Environment", a software system based on the "plug and play" paradigm. Algorithms are encapsulated into specially designed software modules called components which the user can link together produce a simulation. Software components can exist anywhere on a large computing network and yet can be employed to perform a specified simulation on any geographically distributed parallel computing environment to which the user has access. In order to satisfy this design requirement, software components must run as separate threads, respond to signals of other components and, in general, behave quite unlike a standard subroutine library. Therefore, the software architecture must specify the rules for a component to identify standard communication interfaces as well the rules that govern how a component responds to signals. Also, components must be permitted to specify resources and obtain communication lines.

Although the component based "Meta Problem Solving Environment" software paradigm is far more complex than any current software system employed by the biological community today, it has many advantages. By coming together to produce uniform component design protocols, computational biologists could quickly apply the latest algorithmic advances to novel new systems.

Also, simulations could be constructed using the best available components. For example, real space force component from one researcher can be combined with a reciprocal space component from another and an integration component from another group, and could significantly increase computational efficiency compared to any single biological computing package. In addition, with good design protocols, all the components employed in a given simulation need not be built by computational biologists. For instance, parallel FFT component software already exists that could be used to improve significantly SPME performance. Finally, the biological community will not be acting alone in adopting a component based approach. The DOE ASCI project is currently using tools such as CORBA (Common Object Request Broker Architecture), and Los Alamos National Laboratory's PAWS (Parallel Application Workspace) to produce component based software products. Therefore, with appropriate communication within the community and with others, biological computational scientists can begin to produce a new software standard that will function well in large geographically distributed environment and very quickly lead to teraflop computing capability.

ACPI Usability Evaluation and Benchmarking. In the last five years, there has been major progress in benchmarking massively and highly parallel machines for scientific applications. For example, the NAS Parallel Benchmarks (NPB) were a major step toward assessing the computational performance of early MPPs, and served well to compare MPPs with parallel vector machines. Based on numerical algorithms derived from CFD and aerospace applications, the NPB soon were widely adopted as an intercomparison metric of parallel machine performance. Subsequently the NPB were included in the international benchmarking effort, PARKBENCH (PARAllel Kernels and

BENCHmarks). In 1996, portable HPF and MPI versions of the NPB were made available. With increasing industry emphasis on parallelism, a number of other organizations, as part of their procurement process, are designing benchmark suites to evaluate the computational performance of highly parallel systems.

The ACPI program will involve both research and development activities and implementation of comprehensive climate modeling simulations. Therefore, its success will depend not only on raw computational performance but also on an effective environment for code development and analysis of model results. These issues suggest that a new benchmarking approach is needed, one that evaluates not only raw computational performance but also system usability. In addition, there should be formal tracking of benchmark results throughout the course of the program to provide an objective measure of progress. To meet these objectives, ACPI will support benchmarking and performance evaluation in four areas:

- Identify or create scalable, computationally intensive benchmarks that reflect the production computing needs of the particular scientific community or common algorithm, and evaluate the scalability and capability of high performance systems.
- Establish system performance metrics appropriate for scientific modeling.
- Establish usability metrics for assessing the entire research/development/analysis environment.
- Establish a repository of benchmark results for measuring progress in the initiative and for international comparison.

Identify or create scalable, computationally intensive benchmarks that reflect the needs of the scientific community. The purpose of this activity is to develop benchmarks that can be shared by the

community to gain insight into the computational performance of machines that will be proposed by vendors in response to ACPI. It is well known that the GFlops performance level achieved on an application is not a reliable measure of scientific throughput. For example, an algorithm may achieve a very high floating point rate on a certain computer architecture, but may be inefficient at actually solving the problem.

Therefore, accurate and useful benchmarks must be based on a variety of numerical methods that have proven themselves efficient for various modeling problems. Benchmarks for scientific computing applications in general have traditionally used either a functional approach or a performance-oriented approach. In the functional approach, a benchmark represents a workload well if it performs the function as the workload, e.g., a modeling workload is represented by a subset of computational biology codes. In the performance-oriented approach, a benchmark represents a workload well if it exhibits the same performance characteristics as the workload, e.g., a structural genomics modeling workload would be represented by a set of kernels, code fragments, and a communication and I/O test. The assumption is that the system performance can be predicted from an aggregate model of the performance kernels.

We propose that ACPI use mainly a functional approach and identify or develop a set of benchmark models. The benchmark suite also must include a set of simulated coupled models. These benchmarks should be developed in the spirit of the NAS Parallel Benchmarks (the CFD applications benchmarks): the models should exhibit all the important floating point, memory access, communication, and I/O characteristics of a realistic model, yet the code should be scalable, portable, compact, and relatively easy to implement. As mentioned above, this is a substantial effort, because many extant

benchmark codes for some scientific modeling are not readily portable, or are not currently well suited or even capable of running on more than a few tens of processors.

Establish system performance metrics appropriate for scientific modeling. We propose that ACPI select a few full-scale scientific applications that are shared across disciplines, and intercompare parallel systems using the wall clock seconds per model application.

Establish usability metrics and system attributes for assessing the entire research/development/analysis environment. The benchmarks in (1) focus on PE performance alone and do not measure any of the important system features that contribute to the overall utility of a parallel system. These features include code development tools, batch and interactive management capabilities, I/O performance, performance of the memory hierarchy, compiler speed and availability, operating system speed and stability, networking interface performance, mean time between failures or interrupts, etc. Many of these features have been quite weak on current highly parallel computers as compared to vector supercomputers.

We propose that ACPI develop a new set of utility benchmarks and attributes for highly and massively parallel machines as well as clusters of SMPs. The goal of these benchmarks is to derive a set of metrics that measure the overall utility of machines and systems for a productive scientific modeling environment. Such an environment would emphasize model flexibility, rapid job turnaround, and effective capabilities for analysis and data access. Also, the system attributes will specify a set of desired capabilities necessary to administer and manage the system and its resources. At this time there is no industry standard benchmark which addresses these issues. ACPI can drive the state of the art in performance evaluation by initiating such an effort.

Establish a repository of benchmark results for measuring progress in the initiative and for international comparison. Benchmarking codes, metrics, and the detailed benchmark results from the activities in (1), (2), and (3) must be available to the public and the high performance computing community through frequent publication of reports, Web pages, etc., with the goal of influencing the computer industry. ACPI will immediately benefit from using these benchmarks for the next-generation procurement of production machines. ACPI will create a central repository of benchmark codes and performance data. It will also publish annual reports about the measured performance gains

and accomplishments of the initiative, as well as providing international comparison data.

We expect that over the duration of the ACPI, the rapid development of new high-performance computing technology will continue, with rapidly changing architectures and new software and tools becoming available. Hence a benchmark must be designed to be able to evaluate systems across several hardware generations, diverse architectures, and changing software environments. In return, we believe that with a well-designed benchmark, ACPI can drive the HPC industry and create a better focus in the industry on high-end computing.

Appendix 1: Glossary

Aggregation information: The information obtained from applying (a set of) operators to the initial data. For example, monthly sales or expenses, average length of a phone call, or maximum number of calls during a given time.

Amino acid: Any of a class of 20 molecules that are combined to form proteins in living organisms. The sequence of amino acids in a protein and hence protein function are determined by the genetic code.

API (application program interface): An API is a collection of methods that are used by external programs to access and interact with a class or library. Their main function is to provide a consistent interface to a library, isolating programs from changes in the library implementation or functionality.

Automatic Schema Integration: Determination of identical objects represented in the schemata of different databases by programs using a set of built-in rules, without user intervention.

Base pair: A unit of information carried by the DNA molecule. Chemically these are purine and pyrimidine complementary bases connected by weak bonds. Two strands of DNA are held together in a shape of a double helix by the bonds between paired bases.

Bioinformatics: Field of study dealing with management of data in biological sciences.

Change detection: The process of identifying when a data source has changed and how. The types of changes detected include: new data, modifications to existing data, and schema changes.

Chromosome: A self-replicating genetic structure in the cells, containing the cellular DNA that bears in its nucleotide sequence the linear array of genes.

Conflicts: When the same concept is represented in different databases, its

representation may be semantically and syntactically different. For example, a length may be defined in different units, or contain a different set of attributes. These differences are called conflicts, and must be resolved in order to integrate the data from the different sources.

Data cleaning: Removal and/or correction of erroneous data introduced by data entry errors, expired validity of data, or by some other means.

Data ingest: Loading of data into the warehouse.

Data/schema integration: Merging of data/schema from different sources into a single repository, after resolving incompatibilities among the sources.

Data mining: Analysis of raw data to find interesting facts about the data and for purposes of knowledge discovery.

Data warehouse: An integrated repository of data from multiple, possibly heterogeneous data sources, presented with consistent and coherent semantics. Warehouses usually contain summary information represented on a centralized storage facility.

Distributed database: A set of geographically distributed data connected by a computer network and controlled by a single DBMS running at all the sites.

DNA (deoxyribonucleic acid): The molecule that encodes genetic information, a double stranded heteropolymer composed of four types of nucleotides, each containing a different base. See base pair.

Federated databases: An integrated repository data from of multiple, possibly heterogeneous, data sources presented with consistent and coherent semantics. They do not usually contain any summary data, and all

of the data resides only at the data source (i.e. no local storage).

Gene: A fundamental physical and functional unit of heredity. A gene is an ordered sequence of nucleotides located in a particular position on a particular chromosome that encodes a specific functional product (i.e., a protein or RNA molecule). See gene expression.

Gene expression: The process by which a gene's coded information is converted into the structures present and operating in the cell. Expressed genes include those that are transcribed into mRNA and then translated into protein and those that are transcribed into RNA molecules only.

Genetic code: The sequence of nucleotides, coded in triplets along the mRNA, that determines the sequence of amino acids in protein synthesis. The DNA sequence of a gene can be used to specify the mRNA sequence, and the genetic code can in turn be used to specify the amino acid sequence.

Genetic map (linkage map): A map of relative positions of genes or other chromosome markers, determined on the basis of how often they are inherited together. See physical map.

Genome: All the genetic material in the chromosomes of a particular organism; its size is generally given as its total number of base pairs.

Genetic sequencing: The process of identifying the ordered list of amino or nucleic acids that form a particular gene or protein.

Global schema: A schema, or a map of the data content of a data warehouse that integrates the schemata from several source repositories. It is "global", because it is presented to warehouse users as the schema that they can query against to find and relate information from any of the sources, or from the aggregate information in the warehouse.

HGP (human genome project): The U.S. Human Genome Project is a 15-year effort coordinated by the U.S. Department of Energy and the National Institutes of Health to identify all the estimated 100,000 genes in human DNA, determine the sequence of the 3 billion chemical bases that make up human DNA, store this information in databases, and develop tools for data analysis. To help achieve these goals, a study of the genetic makeup of several non-human organisms, including the common human gut bacterium *Escherichia coli*, the fruit fly, and the laboratory mouse is also underway.

Homology: Here, a relationship of having evolved from the same ancestral gene. Two nucleic acids are said to be homologous if their nucleotide sequences are identical or closely related. Similarly, two proteins are homologous if their amino acid sequences are related. Homology can also be inferred from structural similarity.

Intelligent search: Procedures which use additional knowledge to eliminate a majority of space to be searched in order to retrieve a set of data items which satisfy a given set of properties.

JGI (joint genome institute): A massive DOE sponsored project that will integrate human genome research based in three of its national laboratories in California and New Mexico. The Joint Genome Institute is a "virtual laboratory" that will sequence approximately 40 percent of the total human DNA by 2005 and share information through public databases.

Knowledge discovery: The process of identifying and understanding hidden information and patterns in raw data which are of interest to the organization and can be used to improve the business procedures. Data mining is a single, but very important, step in this process.

Metadata: Description of the data. Metadata is used to explicitly associate information with (and knowledge about) data in a repository. This information may range from the simple (it is always an integer) to arbitrarily complex (it is a floating point value representing the temperature of an object in degrees Celsius, and is accurate to 5 significant digits).

Mediated data warehouse architecture: A data warehouse architecture providing access to data from various sources, where not all the data is stored is cached at the warehouse. Mediators or software agents determine the best source of data to satisfy user requests and fetch data from the selected sites in real-time.

Micro-theories: A micro-theory is an ontology about a specific domain, that fits within, and for the most part is consistent with, an ontology with a broader scope. For example, structural biology fits within the larger context of biology. Structural biology will have its own terminology and specific algorithms that apply within the specific domain, but may not be useful or identical to, for example, the genome community.

mRNA (messenger RNA): RNA that serves as a template for protein synthesis. See genetic code.

multi-databases: A repository of data from several, possibly heterogeneous data sources. They do not provide a consistent view of the data contained within the data sources, do not provide summary information, and rarely have a local data store.

nucleic acid: A large molecule composed of nucleotide subunits.

Nucleotide: A subunit of DNA or RNA chemically consisting of a purine or pyrimidine base, a phosphodiester, and a sugar.

object identification: The process of determining data items representing the same real-world concepts in data sources, and

merging the information to a single consistent format. This will allow users to obtain accurate and unique answers to information requests.

object model: A data model for representation real-world entities using the concept of objects which can belong to one of several classes. The classes have a specialization-generalization relationship and contain both data and methods. Objects themselves can be comprised of other objects.

OLAP (on-line analytical processing): Analysis of data along various dimensions (such as time periods, sales areas, and distributors) so as to obtain summary information as well as very specific targeted information to predict business performance based on various parameters.

ontology framework: A framework for data management and manipulation based on a descriptions of inter-relationships of the data items in a specific domain. This allows a more targeted search of the data items using a semantic understanding in addition to using the syntactic values.

PDB (protein data bank): The Protein Data Bank (PDB) is an archive of experimentally determined three-dimensional structures of biological macromolecules, serving a global community of researchers, educators, and students. The archives contain atomic coordinates, bibliographic citations, primary and secondary structure information, as well as crystallographic structure factors and NMR experimental data.

partially materialized views: In a mediated warehouse, a portion of the data represented in the global schema may not be represented in the warehouse, but is only accessible through the mediator. The portion of the data represented locally is said to be materialized. Thus, if there is both local and non-local data represented in the schema, it is said to be partially materialized.

physical map: A map of the locations of identifiable markers on DNA, regardless of inheritance. For the human genome, the lowest-resolution physical map is the banding patterns on the 24 different chromosomes; the highest-resolution map would be the complete nucleotide sequence of the chromosomes. See genetic (linkage) map.

Protein: A large molecule composed of one or more chains of amino acids in a specific order; the order is determined by the sequence of nucleotides in the gene coding for the protein. Proteins are required for the structure, function, and regulation of the body's cells, tissues, and organs, and each protein has unique functions.

protein sequence: The ordered list of amino acids that make up the protein.

protein taxonomy: Proteins can be grouped according to several criteria including their sequence or structural similarity with other proteins. These groupings are often referred to as taxonomies.

RNA (ribonucleic acid): A molecule chemically similar to DNA, involved in the readout of information stored in DNA.

schema (pl. schemata): A description of the data represented within a database. The format of the description varies but includes a table layout for a relational database or an entity-relationship diagram.

SCOP: A database of distinct protein structures and their structural classifications, along with detailed information about the close relatives of any particular protein.

Sequencing: Determination of the order of nucleotides in a DNA or RNA molecule, or the order of amino acids in a protein molecule.

SWISS-PROT: is a protein sequence database with high level of annotations (such as the description of the function of a protein, its domains structure, post-translational modifications, variants, etc.), and a minimal level of redundancy.

terabyte: A measurement of the amount of computer storage. Equivalent to 1,000 gigabytes or 1,000,000 megabytes of information.

unstructured data: Refers to data whose structure is not well defined or strictly enforced. For example, web pages and flat files are unstructured data while a relational database is structured.